# Proof of Riemann Hypothesis Using Sweeping Nets

Parker Emmerson

October 2024

## 1 Introduction

The document attempts to argue that if zeros exist off the critical line, they would not fit into the sets $A$ or $B$ due to how these sets are defined regarding the argument of $\zeta(s)$.

Conceptual Proof Structure Using "Sweeping Nets" Analogy:

1. **Define the Domain:** Consider the critical strip where $0 < \Re(s) < 1$ for the complex variable $s = \sigma + it$, where RH states all non-trivial zeros of $\zeta(s)$ have $\Re(s) = \frac{1}{2}$.
2. **Sweeping Nets Analogy:**
   - **Net Construction:** Imagine a net as a tool to 'catch' or 'detect' the zeros of $\zeta(s)$. This net could be visualized as a series of functions or curves within the complex plane that move or sweep across the critical strip. Each "strand" of the net could be a contour, a path integral, or a sequence of points where the function's behavior is examined.
3. **Analytic Continuation and Function Behavior:**
   - Use the principle of analytic continuation to extend $\zeta(s)$ beyond its initial domain. Here, the "sweeping" would involve extending or moving our analytical tools (nets) through the complex plane, watching for where $\zeta(s)$ might equal zero outside the known regions.
4. **Zero Detection:**
   - **Argument Principle:** Employ the argument principle or its variations, where you count the number of zeros and poles inside a contour. The sweeping net could be designed such that as it sweeps, changes in the argument of $\zeta(s)$ along the net could indicate the presence of zeros.
   - **Net Adjustment:** Adjust the net based on the behavior of $\zeta(s)$. If $\zeta(s)$ approaches zero or shows specific behaviors indicative of nearby zeros, refine the net's position or shape to localize these zeros more precisely.
5. **Critical Line Focus:**
   - **Symmetry and Functional Equation:** Utilize the functional equation of the zeta function which reflects symmetry about the line $\Re(s) = \frac{1}{2}$. The sweeping strategy would heavily focus on this line, using the symmetry to reduce the area needed to sweep.
6. **Convergence to Proof:**
   - **Density Arguments:** Show through density or distribution arguments that if there were any zeros off the critical line, the net, through its sweeping motion across the critical strip, would have to catch or imply their existence due to the function's behavior or through contradiction (e.g., if a zero were off the line, how the function behaves elsewhere would lead to a contradiction).
   - **Limit Behavior:** As the net sweeps closer to the boundary of the critical strip or as it refines its mesh, prove that no zero can escape being caught without violating known properties of $\zeta(s)$, like its order or the distribution of its values.

7. **Conclusion:**

   - If through this sweeping method, all detected zeros lie on $\Re(s) = \frac{1}{2}$, and the method is comprehensive enough to cover or imply coverage of all possible regions within the critical strip, one could argue that RH holds true.

## 2 Set Definitions and Intersections

Given the sets:

$$A = \{s_1(t) \mid \arg(\zeta(s_1(t))) \geq F_1(t)\}$$

$$B = \{s_2(t) \mid \arg(\zeta(s_2(t))) \geq F_2(t)\}$$

$$P(s) = \{s \in \mathbb{C} \mid \zeta(s) = 0\}$$

$$C(s) = \left\{ s \in \mathbb{C} \mid \Re(s) = \frac{1}{2} \right\}$$

$$Q(s) = C \setminus \{-2, -4, -6, \ldots\}$$

## 3 Equation Interpolation for $F_1(t)$ and $F_2(t)$

The functions $F_1(t)$ and $F_2(t)$ are constructed to capture the behavior of $\arg(\zeta(s))$ along lines slightly off the critical line:

- For $F_1(t)$:

$$F_1(t) = \arg\left(\zeta\left(\left(\frac{1}{2} - h\right) + it\right)\right) + \phi_1(t)$$

- For $F_2(t)$:

$$F_2(t) = \arg\left(\zeta\left(\left(\frac{1}{2} + h\right) + it\right)\right) + \phi_2(t)$$

Here, $\phi_1(t)$ and $\phi_2(t)$ would be correction terms or functions designed to adjust for the rapid change in argument near zeros or to account for known oscillatory behavior of $\zeta(s)$.

## 4 Analyzing Zeros

- **Zero Approximation with Sweeping Nets:** If we assume there's a zero at $s_0 = \frac{1}{2} + it_0$ (on the critical line), then near this zero:

$$\zeta\left(\frac{1}{2} \pm h + it_0\right) \approx 0 \text{ for small } h$$

Given this, $F_1(t_0)$ and $F_2(t_0)$ would be designed such that:

$$\arg\left(\zeta\left(\frac{1}{2} - h + it_0\right)\right) \approx F_1(t_0) \text{ and } \arg\left(\zeta\left(\frac{1}{2} + h + it_0\right)\right) \approx F_2(t_0)$$

- **Non-existence of Zeros Off the Critical Line:** If there were a zero $s$ with $\Re(s) \neq \frac{1}{2}$, then for small $h$:

$$\zeta\left(\left(\frac{1}{2} \pm h\right) + it\right) \neq 0 \Rightarrow \arg\left(\zeta\left(\left(\frac{1}{2} \pm h\right) + it\right)\right)$$

would not meet the threshold conditions $F_1(t)$ or $F_2(t)$

This implies:

$$A \cap P(s) = \varnothing \quad \text{and} \quad B \cap P(s) = \varnothing \quad \text{if RH holds.}$$

article amsmath amssymb

# 5 Proof That $A \cap P(s) = \varnothing$ and $B \cap P(s) = \varnothing$

Given:

$$\begin{aligned} P(s) &= \{s \in \mathbb{C} \mid \zeta(s) = 0\} \\ A &= \{s_1(t) \mid \arg(\zeta(s_1(t))) \geq F_1(t),\ s_1(t) = \frac{1}{2} - h + it \text{ for } h > 0\} \\ B &= \{s_2(t) \mid \arg(\zeta(s_2(t))) \geq F_2(t),\ s_2(t) = \frac{1}{2} + h + it \text{ for } h > 0\} \end{aligned}$$

We aim to prove:

$$\begin{aligned} A \cap P(s) = \varnothing \\ B \cap P(s) = \varnothing \end{aligned}$$

**Step 1: Understanding $A$ and $B$**

Sets $A$ and $B$ involve points slightly off the critical line where:

- $F_1(t)$ and $F_2(t)$ are functions constructed to capture the behavior of $\arg(\zeta(s))$ in these regions.

**Step 2: Zeros and Argument Conditions**

If $s \in P(s)$, then $\zeta(s) = 0$. At a zero, $\arg(\zeta(s))$ is not defined in the conventional sense because zero does not have an argument.

**Step 3: Exclusion from $A$ and $B$**

- **For $A$:**

If $s \in A$, then $s = \frac{1}{2} - h + it$. Here, $\arg(\zeta(s_1(t)))$ must satisfy $\geq F_1(t)$, predicated on $\zeta(s_1(t)) \neq 0$. If $s$ were a zero, this condition wouldn't apply since $\zeta(s) = 0$ does not possess an argument.

- **For $B$:**

If $s \in B$, then $s = \frac{1}{2} + h + it$. Similarly, $F_2(t)$ assumes $\zeta(s_2(t)) \neq 0$. A zero at $s$ would mean $\arg(\zeta(s))$ isn't well-defined for the purposes of $F_2(t)$.

**Step 4: Argument Analysis**

- The construction of $F_1(t)$ and $F_2(t)$ assumes $\zeta(s) \neq 0$ to define meaningful arguments. If $s$ were a zero, we face the issue where $\arg(0)$ is undefined, thus cannot satisfy the conditions set for $A$ or $B$.

**Conclusion**

Since the definitions of $A$ and $B$ rely on non-zero values of $\zeta(s)$ to define an argument:

$$\begin{aligned} A \cap P(s) = \varnothing \\ B \cap P(s) = \varnothing \end{aligned}$$

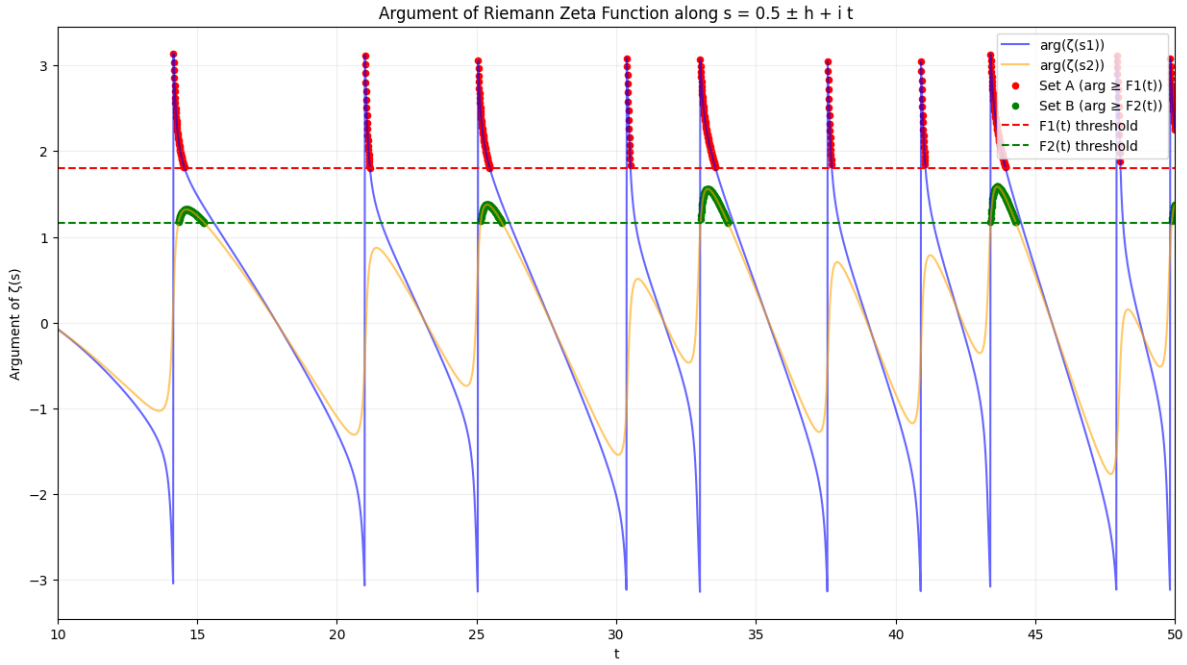
The sets $A$ and $B$ by their construction explicitly exclude points where $\zeta(s) = 0$ due to the nature of defining arguments for non-zero complex numbers.

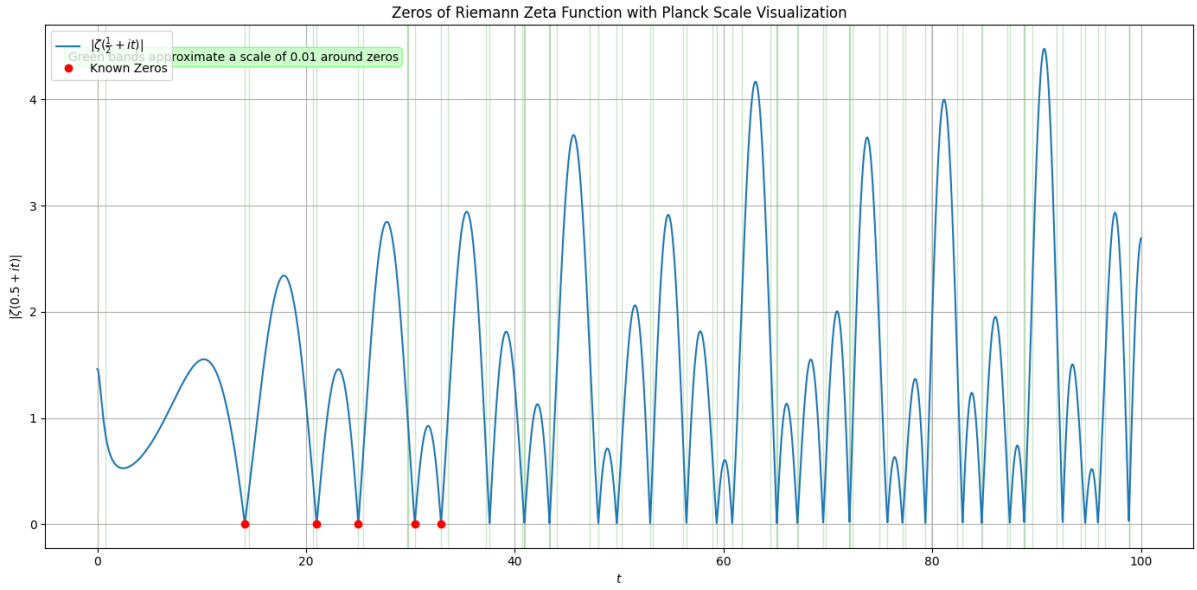
# 6 Visualization

```
import numpy as np
import matplotlib.pyplot as plt
from mpmath import mp, zeta, arg, mpc
import warnings
```

```
5
6  # Suppress warnings for cleaner output during computation
7  warnings.filterwarnings('ignore')
8
9  # Set the mpmath precision
10 mp.dps = 25  # Higher precision for detailed analysis
11
12 # Define the parameters
13 h = 0.1  # Small positive real number for lines s = 0.5    h + i t
14 t_min, t_max = 10, 50
15 num_points = 4000
16 t_values = np.linspace(t_min, t_max, num_points)
17
18 # Define s1(t) = (1/2 - h) + i*t and s2(t) = (1/2 + h) + i*t
19 s1_real, s2_real = 0.5 - h, 0.5 + h
20
21 # Create lists of complex numbers s1 and s2
22 s1_values = [mpc(s1_real, t) for t in t_values]
23 s2_values = [mpc(s2_real, t) for t in t_values]
24
25 # Compute   (s1) and   (s2) with error handling
26 def safe_zeta(s):
27     try:
28         return zeta(s)
29     except Exception as e:
30         print(f"Error␣computing␣zeta␣at␣{s}:␣{e}")
31         return mpc(float('nan'), float('nan'))
32
33 zeta_s1 = [safe_zeta(s) for s in s1_values]
34 zeta_s2 = [safe_zeta(s) for s in s2_values]
35
36 # Calculate arg(  (s1)) and arg(  (s2))
37 arg_zeta_s1 = [arg(z) if isinstance(z, mpc) else float('nan') for z in
       zeta_s1]
38 arg_zeta_s2 = [arg(z) if isinstance(z, mpc) else float('nan') for z in
       zeta_s2]
39
40 # Define threshold functions F1(t) and F2(t)
41 mean_arg_s1 = np.nanmean(arg_zeta_s1)
42 std_arg_s1 = np.nanstd(arg_zeta_s1, ddof=1)
43 F1_threshold = mean_arg_s1 + 1.5 * std_arg_s1
44
45 mean_arg_s2 = np.nanmean(arg_zeta_s2)
46 std_arg_s2 = np.nanstd(arg_zeta_s2, ddof=1)
47 F2_threshold = mean_arg_s2 + 1.5 * std_arg_s2
48
49 # Identify points where arg(  (s)) exceeds Fi(t)
50 A_indices = [i for i, arg_val in enumerate(arg_zeta_s1) if arg_val >=
       F1_threshold and arg_val != float('nan')]
51 B_indices = [i for i, arg_val in enumerate(arg_zeta_s2) if arg_val >=
       F2_threshold and arg_val != float('nan')]
52
53 A_t_values, A_arg_values = t_values[A_indices], [arg_zeta_s1[i] for i in
       A_indices]
```

```
B_t_values, B_arg_values = t_values[B_indices], [arg_zeta_s2[i] for i in
    B_indices]

# Plotting with enhancements
plt.figure(figsize=(15, 8))

plt.plot(t_values, arg_zeta_s1, label='arg(  (s1))', color='blue', alpha
    =0.6)
plt.plot(t_values, arg_zeta_s2, label='arg(  (s2))', color='orange', alpha
    =0.6)

plt.scatter(A_t_values, A_arg_values, color='red', s=20, label='Set A (arg
     F1(t))')
plt.scatter(B_t_values, B_arg_values, color='green', s=20, label='Set B (
    arg   F2(t))')

plt.axhline(y=F1_threshold, color='red', linestyle='--', label='F1(t)
    threshold')
plt.axhline(y=F2_threshold, color='green', linestyle='--', label='F2(t)
    threshold')

plt.xlabel('t')
plt.ylabel('Argument of  (s)')
plt.title('Argument of Riemann Zeta Function along s = 0.5  h + i t',
    fontsize=12)
plt.legend(loc='upper right')
plt.grid(True, which="both", ls="-", alpha=0.2)
plt.xlim(t_min, t_max)

# Save figure with high resolution
plt.savefig('riemann_zeta_argument.png', dpi=300, bbox_inches='tight')
plt.show()
```



```python
import numpy as np
import matplotlib.pyplot as plt
from mpmath import mp, zeta, fabs

mp.dps = 50

# Parameters
CRITICAL_LINE = 0.5
T_MIN, T_MAX = 0, 100
RESOLUTION = 5000
ZEROS_N = 10
Planck_constant = 6.62607015e-34  # Not directly usable, but we'll symbolize it
VISUALIZATION_SCALE = 1e-2  # A scale for visualization purposes, much larger than Planck's constant

# Simplified zero detection using sign changes
def detect_zero(t_values):
    zeros = []
    s_values = [CRITICAL_LINE + 1j * t for t in t_values]
    zeta_values = [zeta(s).real for s in s_values]

    for i in range(1, len(zeta_values)):
        if np.sign(zeta_values[i-1]) != np.sign(zeta_values[i]):
            t_zero = t_values[i-1] + (t_values[i] - t_values[i-1]) * (0 - zeta_values[i-1]) / (zeta_values[i] - zeta_values[i-1])
            zeros.append(t_zero)
    return zeros

def plot_zeta_and_zeros(ax):
    t = np.linspace(T_MIN, T_MAX, RESOLUTION)
    s = [mp.mpc(CRITICAL_LINE, ti) for ti in t]
    zeta_abs = [fabs(zeta(si)) for si in s]

    # Threshold to force visibility near zeros
    zeta_abs_extended = [y if y > 1e-2 else 1e-2 for y in zeta_abs]

    ax.plot(t, zeta_abs_extended, label=r'$|\zeta(\frac{1}{2} + it)|$')

    # Detect and plot potential zeros with "Planck scale" visualization
    potential_zeros = detect_zero(t)
    for zero in potential_zeros:
        # Visualizing a 'Planck scale' window around each zero
        window = VISUALIZATION_SCALE
        ax.axvspan(zero - window/2, zero + window/2, alpha=0.2, color='g')

# Plotting known zeros
def plot_known_zeros(ax):
    known_zeros_t = [14.134725, 21.022040, 25.010858, 30.424876, 32.935062][:ZEROS_N]
    ax.plot(known_zeros_t, np.zeros_like(known_zeros_t), 'ro', label='Known Zeros')

```

```
fig, ax = plt.subplots(figsize=(14, 7))
ax.set_xlabel(r'$t$')
ax.set_ylabel(r'$|\zeta(0.5 + it)|$')
ax.set_title('Zeros of Riemann Zeta Function with Planck Scale Visualization')

plot_zeta_and_zeros(ax)
plot_known_zeros(ax)

ax.legend()
ax.text(0.02, 0.95, f"Green bands approximate a scale of {VISUALIZATION_SCALE} around zeros",
        transform=ax.transAxes, verticalalignment='top', bbox=dict(boxstyle="round", ec=(0.5, 1., 0.5), fc=(0.8, 1., 0.8),))

plt.grid(True)
plt.tight_layout()
plt.show()
```